CLAIMS

What is claimed is:

1. A method comprising:

storing an architectural state of a processor corresponding to a first checkpoint;

storing non-deterministic events that occur subsequent to the storage of the first checkpoint;

determining whether an processing error has occurred subsequent to the storage of the first checkpoint; and

restoring the architectural state of the processor corresponding to the first checkpoint and re-executing the non-deterministic events if a processing error is detected.

2. The method of claim 1 wherein storing the architectural state of the processor corresponding to the first checkpoint comprises:

synchronizing a leading thread of instructions and a trailing thread of instructions;

checking outputs from the leading thread of instructions and the trailing thread of instructions for faults; and

storing values from one or more architectural registers in a memory external to one or more processors executing the leading thread of instructions and the trailing thread of instructions.

3. The method of claim 2 wherein the leading thread of instructions and the trailing thread of instructions are executed by a single processor.

4. The method of claim 2 wherein the leading thread of instructions and the trailing thread of instructions are executed by multiple processors.

5. The method of claim 1 wherein restoring the architectural state of the processor corresponding to the first checkpoint and re-executing the non-deterministic events if a processing error is detected comprises:

selectively flushing results of instructions that started execution after an instruction causing the fault started execution;

restoring architectural state to a checkpoint corresponding to a state at which the instruction causing the fault started execution; and

re-executing instructions executed after the checkpoint to the instruction causing the fault using the logged non-deterministic events.

6. The method of claim 5 further comprising discarding one or more outputs generated by the re-execution of the instructions executed after the checkpoint to the instruction causing the fault using the logged non-deterministic events.

7. The method of claim 5 further comprising continuing execution of instructions subsequent to the instruction causing the fault.

8. The method of claim 5 wherein restoring architectural state to a checkpoint corresponding to a state at which the instruction causing the fault started

execution comprises restoring the architectural state to a state prior to execution of the instruction causing the fault.

9. The method of claim 5 wherein restoring architectural state to a checkpoint corresponding to a state at which the instruction causing the fault started execution comprises restoring the architectural state to a state a the beginning of execution of the instruction causing the fault.

10. The method of claim 1 wherein the non-deterministic event comprises a load operation.

11. The method of claim 1 wherein the non-deterministic event comprises an interrupt.

12. The method of claim 11 wherein the interrupt comprises an asynchronous interrupt.

13. The method of claim 1 wherein the non-deterministic event comprises a timing-dependent operation.

14. The method of claim 13 wherein the timing-dependent operation comprises a read operation of a cycle counter.

15. An apparatus comprising:

means for storing an architectural state of a processor corresponding to a first checkpoint;

means for storing non-deterministic events that occur subsequent to the storage of the first checkpoint;

means for determining whether an processing error has occurred subsequent to the storage of the first checkpoint; and

means for restoring the architectural state of the processor corresponding to the first checkpoint and re-executing the non-deterministic events if a processing error is detected.

16. The apparatus of claim 15 wherein the means for restoring the architectural state of the processor corresponding to the first checkpoint and re-executing the non-deterministic events if a processing error is detected comprises:

means for selectively flushing results of instructions that started execution after an instruction causing the fault started execution;

means for restoring architectural state to a checkpoint corresponding to a state at which the instruction causing the fault started execution; and

means for re-executing instructions executed after the checkpoint to the instruction causing the fault using the logged non-deterministic events.

17. The apparatus of claim 15 further comprising means for discarding one or more outputs generated by the re-execution of the instructions executed after the checkpoint to the instruction causing the fault using the logged non-deterministic events.

18. An apparatus comprising:

leading thread execution circuitry to execute a leading thread of instructions;

trailing thread execution circuitry to execute a trailing thread of instructions; and

a memory coupled with the leading thread execution circuitry and the trailing thread execution circuitry to store information related to non-deterministic events, wherein the information related to non-deterministic events is stored at least until a subsequent checkpoint is validated.

19. The apparatus of claim 18 wherein the memory stores a load value queue that stores replication entries corresponding to load operations executed by the leading thread execution circuitry and not executed by the trailing thread execution circuitry, and further wherein the load value queue stores recovery entries corresponding to load operations executed by the leading thread execution circuitry and by the trailing thread execution circuitry and not incorporated into information corresponding to a checkpoint.

20. The apparatus of claim 19 wherein the load value queue further stores information corresponding to one or more of an interrupt and a timing-dependent operation.

21. The apparatus of claim 18 wherein the memory stores a first load value queue that stores replication entries corresponding to load operations executed by the leading thread execution circuitry and not executed by the trailing thread execution circuitry, and a second load value queue stores recovery entries corresponding to load operations executed by the leading thread execution circuitry and by the trailing thread execution circuitry and not incorporated into information corresponding to a checkpoint.

22. The apparatus of claim 21 wherein the first load value queue and the second load value queue further store information corresponding to one or more of an interrupt and a timing-dependent operation.

23. A system comprising:

leading thread execution circuitry to execute a leading thread of instructions;

trailing thread execution circuitry to execute a trailing thread of instructions;

a memory controller coupled with the leading thread execution circuitry; and

a memory coupled with the leading thread execution circuitry and the trailing thread execution circuitry to store information related to non-deterministic events, wherein the information related to non-deterministic events is stored at least until a subsequent checkpoint is validated.

24. The system of claim 23 wherein the memory stores a load value queue that stores replication entries corresponding to load operations executed by the leading thread execution circuitry and not executed by the trailing thread execution circuitry, and further

wherein the load value queue stores recovery entries corresponding to load operations executed by the leading thread execution circuitry and by the trailing thread execution circuitry and not incorporated into information corresponding to a checkpoint.

25. The system of claim 24 wherein the load value queue further stores information corresponding to one or more of an interrupt and a timing-dependent operation.

26. The system of claim 23 wherein the memory stores a first load value queue that stores replication entries corresponding to load operations executed by the leading thread execution circuitry and not executed by the trailing thread execution circuitry, and a second load value queue stores recovery entries corresponding to load operations executed by the leading thread execution circuitry and by the trailing thread execution circuitry and not incorporated into information corresponding to a checkpoint.

27. The system of claim 26 wherein the first load value queue and the second load value queue further store information corresponding to one or more of an interrupt and a timing-dependent operation.